

AMENDMENTS

In the Claims

Please cancel claims 9, 31-32, and 53-54 without prejudice.

Please amend claims 1, 8, 12, 28, 33, and 49 as shown herein.

Claims 1-8, 10, 12-30, 33-37, 42-52, and 55-70 are pending and are listed following:

1. (currently amended) A finite state model-based testing system comprising:

a model generation engine configured to generate a model of a software application, the model being generated from parameters that describe the software application;

a user interface to enable user entry of the parameters to define the model;

a graph traversal program to generate a test sequence of inputs for the software application, the test sequence of inputs being generated from the model of the software application; and

a test driver to initiate a test of the software application with a the test sequence of inputs ~~generated from the model of the software application.~~

1 **2. (previously presented)** A finite state model-based testing system
2 as recited in claim 1, wherein the user interface enables a user to enter state
3 information and transition information that describes the software application, the
4 transition information describing a next state of the software application after an
5 input has been applied to a current state of the software application.

6
7 **3. (previously presented)** A finite state model-based testing system
8 as recited in claim 1, wherein the user interface enables a user to enter state
9 information that describes the software application, the state information
10 comprising:

11 an operational mode of the software application, wherein the operational
12 mode is an attribute of a particular state of the software application;

13 at least one modal value that describes a behavior of the operational mode;

14 and

15 an input of the software application.

16
17 **4. (previously presented)** A finite state model-based testing system
18 as recited in claim 1, wherein the user interface enables a user to enter transition
19 information that describes the software application, the transition information
20 comprising:

21 a current state of the software application, the current state being associated
22 with an input of the software application; and

23 a next state that indicates the state of the software application after the input
24 has been applied to the current state of the software application.
25

1
2 **5. (previously presented)** A finite state model-based testing system
3 as recited in claim 1, wherein the user interface comprises a model editor to enable
4 user entry of an operational mode, a modal value associated with the operational
5 mode, and an input of the software application to define the model.
6

7 **6. (previously presented)** A finite state model-based testing system
8 as recited in claim 1, wherein the user interface comprises a rules editor to enable
9 user entry of an input of the software application, a current state of the software
10 application, and a next state of the software application to indicate the state of the
11 software application after the input has been applied to the current state to define
12 the model.
13
14
15
16
17
18
19
20
21
22
23
24
25

1 **7. (previously presented)** A finite state model-based testing system
2 as recited in claim 1, wherein the model of the software application is a state table
3 having at least one state table entry, and wherein:

4 a state table entry comprises:

- 5 (1) a current state of the software application;
6 (2) an input of the software application;
7 (3) a next state that indicates the state of the software application
8 after the input has been applied to the current state of the software
9 application; and wherein

10 the model generation engine is further configured to evaluate the current
11 state of the software application to determine if an input of the software
12 application can be applied to the current state and in the event that the input can be
13 applied to the current state, writes a state table entry out to the state table.

14
15 **8. (currently amended)** A finite state model-based testing system
16 as recited in claim 1, wherein the user interface comprises a graph traversal menu
17 to enable a user to select a the graph traversal program and generate the test
18 sequence of inputs for the software application.

19
20 **9. (canceled)**

1 **10. (previously presented)** A finite state model-based testing system
2 as recited in claim 1, wherein the user interface comprises a test execution menu to
3 enable a user to select the test driver and initiate the test of the software
4 application.

5
6 **11. (canceled)**

7
8 **12. (currently amended)** A testing interface for testing a software
9 application, the testing interface comprising:

10 a model editor to enable user entry of state information to define a model of
11 the software application;

12 a rules editor to enable user entry of transition information to further define
13 the model of the software application, the transition information describing a next
14 state of the software application after an input has been applied to a current state
15 of the software application; and

16 a model generation engine configured to generate the model of a software
17 application from the state information and the transition information; and

18 a menu configured to enable user selection of a graph traversal program to
19 generate a test sequence of inputs for the software application from the model of
20 the software application, the menu further configured to enable user selection of a
21 test driver to initiate a test of the software application with the test sequence of
22 inputs.

1 **13. (previously presented)** A testing interface as recited in claim 12,
2 further comprising a graphical user interface that includes the model editor and the
3 rules editor.

4
5 **14. (previously presented)** A testing interface as recited in claim 12,
6 wherein the state information comprises:

7 an operational mode of the software application which is an attribute of a
8 particular state of the software application;

9 at least one modal value that describes a behavior of the operational mode;
10 and

11 an input of the software application.

12
13 **15. (previously presented)** A testing interface as recited in claim 12,
14 wherein the transition information comprises:

15 a current state of the software application, the current state being associated
16 with an input of the software application; and

17 a next state of the software application that indicates the state of the
18 software application after the input has been applied to the current state of the
19 software application.

1 **16. (previously presented)** A testing interface as recited in claim 12,
2 wherein the model editor comprises:

3 an operational modes entry field to enable user entry of an operational
4 mode of the software application; and

5 an operational modes list field to display the operational mode.

6
7 **17. (previously presented)** A testing interface as recited in claim 12,
8 wherein the model editor comprises:

9 a modal values entry field to enable user entry of a modal value associated
10 with an operational mode of the software application; and

11 a modal values list field to display the modal value.

12
13 **18. (previously presented)** A testing interface as recited in claim 12,
14 wherein the model editor comprises:

15 an inputs entry field to enable user entry of an input of the software
16 application; and

17 an inputs list field to display the input of the software application.

1 **19. (previously presented)** A testing interface as recited in claim 12,
2 wherein the rules editor comprises fields to display the state information that can
3 be entered using the model editor, the fields comprising:

4 inputs fields to display inputs of the software application, wherein the
5 inputs fields also enable user selection of an input of the software application;

6 operational modes fields to display operational modes of the software
7 application, wherein the operational modes fields also enable user selection of an
8 operational mode; and

9 modal values fields to display modal values associated with an operational
10 mode, wherein the modal values fields also enable user selection of a modal value.
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

1 **20. (previously presented)** A testing interface as recited in claim 19,
2 wherein the rules editor enables user entry of the transition information, and
3 wherein:

4 the transition information comprises:

5 a current state of the software application, the current state being
6 associated with the input of the software application;

7 a next state that indicates the state of the software application after
8 the input has been applied to the current state of the software application;

9 the rules editor comprises:

10 an inputs field to enable user entry of the input;

11 a current state operational mode field to enable user entry of the
12 operational mode as a current state operational mode;

13 a current state modal value field to enable user entry of the modal
14 value associated with the current state operational mode;

15 a next state operational mode field to enable user entry of the
16 operational mode as a next state operational mode; and

17 a next state modal value field to enable user entry of the modal value
18 associated with the next state operational mode.

1 **21. (previously presented)** A testing interface as recited in claim 12,
2 wherein the model is a state table having at least one state table entry, the state
3 table entry including a current state of the software application, an input of the
4 software application, and a next state of the software application that indicates the
5 state of the software application after the input has been applied to the current state
6 of the software application; and wherein

7 the model editor enables user initiation of the model generation engine to
8 generate the model of the software application, the model generation engine being
9 further configured to evaluate a current state of the software application to
10 determine if an input of the software application can be applied to the current state
11 and in the event that the input can be applied to the current state, write a state table
12 entry out to the state table.

13
14 **22. (previously presented)** A testing interface as recited in claim 12,
15 wherein the model editor enables user initiation of a graph traversal program to
16 generate a test sequence of inputs for the software application.

17
18 **23. (previously presented)** A testing interface as recited in claim 12,
19 wherein the testing interface further comprises a graph traversal menu to enable
20 user selection of a graph traversal program to generate a test sequence of inputs for
21 the software application.

1 **24. (previously presented)** A testing interface as recited in claim 23,
2 wherein the graph traversal menu comprises:

3 a graph traversal program field to enable user selection of the graph
4 traversal program;

5 a model file field to enable user selection of the model of the software
6 application to be tested; and

7 a test sequence file field to enable user entry of a memory storage location
8 for a test sequence file, the test sequence file containing the test sequence of inputs
9 for the software application.
10

11 **25. (previously presented)** A testing interface as recited in claim 12,
12 wherein the model editor enables user initiation of a test driver program to read a
13 test sequence of inputs for the software application and apply the test sequence of
14 inputs to the software application.
15

16 **26. (previously presented)** A testing interface as recited in claim 12,
17 wherein the testing interface further comprises a test execution menu to enable
18 user selection of a test driver program to read a test sequence of inputs for the
19 software application and apply the test sequence of inputs to the software
20 application.
21
22
23
24
25

1 **27. (previously presented)** A testing interface as recited in claim 26,
2 wherein the test execution menu comprises:

3 a test driver program field to enable user selection of the test driver
4 program;

5 a test sequence file field to enable user selection of a test sequence file
6 containing the test sequence of inputs for the software application; and

7 a test monitoring interval field to enable user entry of a timing interval to
8 define how often the test driver program can be monitored to detect a failure of the
9 test driver program.

10
11 **28. (currently amended)** A user interface to enable user entry of
12 parameters to define a model of a software application to be tested, the user
13 interface comprising:

14 an operational modes field to enable user entry of an operational mode of
15 the software application, the operational mode being an attribute of a particular
16 state of the software application; ~~and~~

17 a modal values field to enable user entry of at least one modal value
18 associated with the operational mode, the modal value describing a behavior of the
19 operational mode; and

20 a menu configured to enable user initiation of a graph traversal program to
21 generate a test sequence of inputs for the software application, the menu further
22 configured to enable user initiation of a test driver to apply the test sequence of
23 inputs to the software application to test the software application.

1 **29. (original)** A user interface as recited in claim 28, further
2 comprising an input field to enable user entry of an input of the software
3 application.

4
5 **30. (original)** A user interface as recited in claim 29, further
6 comprising:

7 an operational modes list field to display the operational mode;

8 a modal values list field to display the modal value; and

9 an inputs list field to display the input of the software application.

10
11 **31. (canceled)**

12 **32. (canceled)**
13
14
15
16
17
18
19
20
21
22
23
24
25

1 **33. (currently amended)** A user interface to enable user entry of
2 parameters to define a model of a software application to be tested, the user
3 interface comprising:

4 an inputs field to enable user entry of an input of the software application;

5 current state fields to enable user entry of a current state of the software
6 application, the current state being associated with the input; and

7 next state fields to enable user entry of a next state of the software
8 application, the next state indicating the state of the software application after the
9 input has been applied to the current state of the software application.

10 a menu configured to enable generation of a test sequence of inputs from
11 the parameters that define the model of the software application, the menu further
12 configured to enable application of the test sequence of inputs to the software
13 application to test the software application.

14
15 **34. (original)** A user interface as recited in claim 33, further
16 comprising a rules field to enable user entry of a rule to describe a transition of the
17 software application from a current state to a next state.

18
19 **35. (original)** A user interface as recited in claim 34, further
20 comprising a control to enable a user to disable a rule such that the model of the
21 software application will be defined without the rule.

1 **36. (original)** A user interface as recited in claim 33, wherein:

2 the current state fields include:

3 a current state operational mode field to enable user entry of a
4 current state operational mode of the software application;

5 a current state modal value field to enable user entry of at least one
6 current state modal value associated with the current state operational
7 mode;

8 the next state fields include:

9 a next state operational mode field to enable user entry of a next
10 state operational mode of the software application; and

11 a next state modal value field to enable user entry of at least one next
12 state modal value associated with the next state operational mode.

1 **37. (original)** A user interface as recited in claim 36, wherein:

2 the current state fields include:

3 a current state relational operator field to indicate the current state
4 modal value relation to the current state operational mode;

5 a current state concatenation operator field to indicate the relation
6 between a first current state rule criteria and a second current state rule
7 criteria.

8 the next state fields include:

9 a next state relational operator field to indicate the next state modal
10 value relation to the next state operational mode; and

11 a next state concatenation operator field to indicate the relation
12 between a first next state rule criteria and a second next state rule criteria.

13
14 **38-41. (canceled)**

1 **42. (previously presented)** A finite state model-based testing system
2 comprising:

3 a model editor to enable user entry of state information to define a model of
4 a software application to be tested;

5 a rules editor to enable user entry of transition information to further define
6 the model of the software application, the transition information describing a next
7 state of the software application after an input has been applied to a current state
8 of the software application;

9 a model generation engine to generate the model of the software
10 application, the model being generated from the state information and the
11 transition information;

12 a graph traversal menu to enable user selection of a graph traversal program
13 to generate a test sequence of inputs for the software application, the test sequence
14 of inputs being generated from the model of the software application; and

15 a test execution menu to enable user selection of a test driver to read the test
16 sequence of inputs and apply the test sequence of inputs to the software
17 application.
18
19
20
21
22
23
24
25

1 **43. (original)** A finite state model-based testing system as recited in
2 claim 42, wherein the model editor comprises:

3 operational modes fields to enable user entry of an operational mode of the
4 software application;

5 modal values fields to enable user entry of a modal value associated with
6 the operational mode; and

7 inputs fields to enable user entry of an input of the software application.
8

9 **44. (original)** A finite state model-based testing system as recited in
10 claim 42, wherein the rules editor comprises fields to display the state information
11 that can be entered using the model editor, the fields comprising:

12 inputs fields to display inputs of the software application, wherein the
13 inputs fields also enable user selection of an input of the software application;

14 operational modes fields to display operational modes of the software
15 application, wherein the operational modes fields also enable user selection of an
16 operational mode; and

17 modal values fields to display modal values associated with an operational
18 mode, wherein the modal values fields also enable user selection of a modal value.
19
20
21
22
23
24
25

1 **45. (previously presented)** A finite state model-based testing system
2 as recited in claim 42, wherein the model is a state table having at least one state
3 table entry, the state table entry including a current state of the software
4 application, an input of the software application, and a next state of the software
5 application; and wherein

6 the model editor enables user initiation of the model generation engine
7 which is configured to evaluate a current state of the software application to
8 determine if an input of the software application can be applied to the current state
9 and in the event that the input can be applied to the current state, write a state table
10 entry out to the state table.

11
12 **46. (original)** A finite state model-based testing system as recited in
13 claim 42, wherein the model editor facilitates user initiation of the rules editor.

14
15 **47. (original)** A finite state model-based testing system as recited in
16 claim 42, wherein the model editor facilitates user initiation of the graph traversal
17 menu.

18
19 **48. (original)** A finite state model-based testing system as recited in
20 claim 42, wherein the model editor facilitates user initiation of the test execution
21 menu.

1 **49. (currently amended)** A computer system comprising:

2 a processor;

3 a memory;

4 a user interface application stored in the memory and executable on the
5 processor to facilitate user definition of a finite-state model to test a software
6 application;

7 the user interface application having computer readable instructions to
8 display a graphical user interface; and

9 a model generation engine stored in memory and executable on the
10 processor to generate the finite-state model of the software application, the finite-
11 state model being generated from state information and transition information that
12 describes the software application;

13 at least one graph traversal program stored in the memory and executable
14 on the processor to generate a test sequence of inputs for the software application,
15 the user interface presenting a graph traversal menu to enable a user to select the
16 graph traversal program; and

17 at least one test driver program stored in the memory and executable on the
18 processor to execute a test sequence of application inputs on the software
19 application, the user interface presenting a test execution menu to enable a user to
20 select the test driver program.

1 **50. (previously presented)** A computer system as recited in claim
2 49, wherein the graphical user interface enables a user to enter the state
3 information and the transition information that describes the software application,
4 the transition information describing a next state of the software application after
5 an input has been applied to a current state of the software application.

6
7 **51. (previously presented)** A computer system as recited in claim
8 49, wherein the user interface comprises a model editor to enable user entry of
9 operational modes, modal values, and inputs of the software application to define
10 the finite-state model.

11 **52. (previously presented)** A computer system as recited in claim
12 49, wherein the user interface comprises a rules editor to enable user entry of an
13 input of the software application, a current state of the software application, and a
14 next state of the software application to define the finite-state model.

15
16 **53. (canceled)**

17 **54. (canceled)**
18
19
20
21
22
23
24
25

1 **55. (original)** A computer system as recited in claim 49, further
2 comprising a data structure stored in the memory, the data structure comprising:

3 at least one mode data structure to hold an operational mode of a software
4 application and at least one modal value associated with the operational mode; and

5 at least one rule data structure to hold an input of the software application, a
6 current state of a software application, and a next state of the software application.

7
8 **56. (previously presented)** A method comprising:

9 presenting a graphical user interface that facilitates user entry of state
10 information and transition information that describes a software application to be
11 tested;

12 generating a model of the software application from the state information
13 and the transition information; and

14 generating a test sequence of inputs for the software application from the
15 model of the software application.

16
17 **57. (previously presented)** A method as recited in claim 56, further
18 comprising presenting a graphical user interface that facilitates user selection of a
19 graph traversal program to generate the test sequence of inputs for the software
20 application.

1 **58. (previously presented)** A method as recited in claim 56, further
2 comprising:

3 presenting a graphical user interface that facilitates user selection of a test
4 driver program; and

5 executing the test sequence of inputs on the software application with the
6 test driver program.

7
8 **59. (original)** A method as recited in claim 56, further comprising
9 enabling a user to define a transition rule of the software application, wherein the
10 enabling comprises presenting a graphical user interface to facilitate user entry of
11 an input of the software application, a current state of the software application
12 associated with the input, and a next state of the software application.

13
14 **60. (original)** A method as recited in claim 56, further comprising:
15 presenting a graphical user interface that facilitates a user defining a
16 transition rule of the software application and disabling the transition rule; and
17 generating the model of the software application without incorporating the
18 disabled transition rule.

1 **61. (original)** A method as recited in claim 56, wherein generating a
2 model of the software application comprises:

3 evaluating a current state of the software application to determine if an
4 input of the software application can be applied to the current state; and

5 in the event that the input can be applied to the current state, writing a state
6 table entry out to a state table.

7
8 **62. (original)** A method as recited in claim 56, further comprising
9 enabling a user to define the state information, wherein the enabling comprises
10 presenting a graphical user interface to facilitate user entry of an operational mode
11 of the software application, at least one modal value associated with the
12 operational mode, and an input of the software application.

13
14 **63. (previously presented)** A method as recited in claim 56, further
15 comprising enabling a user to define the transition information, wherein the
16 enabling comprises presenting a graphical user interface to facilitate user entry of
17 a current state of the software application that is associated with an input of the
18 software application, and user entry of a next state that indicates the state of the
19 software application after the input has been applied to the current state of the
20 software application.

21
22 **64. (original)** A computer-readable medium comprising computer
23 executable instructions that, when executed, direct a computing system to perform
24 the method of claim 56.
25

1 **65. (previously presented)** A method comprising:
2 presenting a user interface that facilitates user entry of state information and
3 transition information about a software application to be tested;
4 initiating, via the user interface, generation of a model of the software
5 application, the model being generated from the state information and the
6 transition information;
7 selecting, via the user interface, a graph traversal program that generates a
8 test sequence of inputs for the software application, the test sequence of inputs
9 being generated from the model of the software application; and
10 selecting, via the user interface, a test driver program that executes a test
11 sequence of inputs on the software application.

12
13 **66. (original)** A computer-readable medium comprising computer
14 executable instructions that, when executed, direct a computing system to perform
15 the method of claim 65.
16
17
18
19
20
21
22
23
24
25

1 **67. (previously presented)** A method comprising:
2 receiving state information about a software application to be tested;
3 receiving transition information about the software application;
4 generating a model of the software application, the model being generated
5 from the state information and the transition information;
6 generating a test sequence of inputs for the software application with a
7 graph traversal program, the test sequence of inputs being generated from the
8 model of the software application; and
9 executing the test sequence of inputs on the software application.

10
11 **68. (original)** A method as recited in claim 67, wherein generating a
12 model of the software application comprises:

13 evaluating a current state of the software application to determine if an
14 input of the software application can be applied to the current state; and

15 in the event that the input can be applied to the current state, writing a state
16 table entry out to a state table.

17
18 **69. (original)** A computer-readable medium comprising computer
19 executable instructions that, when executed, direct a computing system to perform
20 the method of claim 67.

1 **70. (previously presented)** A computer-readable medium
2 comprising computer executable instructions that, when executed, direct a
3 computing system to:

4 receive state information about a software application to be tested;

5 receive transition information about the software application;

6 generate a model of the software application from the state information and
7 the transition information;

8 generate a test sequence of inputs for the software application with a graph
9 traversal program, the test sequence of inputs being generated from the model of
10 the software application; and

11 execute the test sequence of inputs on the software application.